第 3 章　HTML5 拖放 API 项目

本章主要包含两个基于 HTML5 拖放 API 的应用设计实例，一是仿回收站效果的设计与实现，二是图片相框展示的设计与实现。在仿回收站项目中，主要难点为元素的拖曳以及回收效果；在图片相框展示项目中，主要难点为文件的拖曳、自动生成带有相框的图片以及显示图片文件信息的技术。

本章学习目标：

- 学习如何综合应用 HTML5 拖放 API、CSS 与 JavaScript 开发仿回收站效果；
- 学习如何综合应用 HTML5 拖放 API、CSS 与 JavaScript 开发图片相框展示效果。

3.1　仿回收站效果的设计与实现

【例 3-1】 基于 HTML5 拖放 API 的仿回收站效果的设计与实现

背景介绍：在 Windows 等操作系统中均包含回收站功能，用户可以直接将不需要的文件拖曳并放置到桌面回收站图标上以实现文件的删除。

功能要求：使用 HTML5 拖放 API 相关技术在网页上实现仿回收站的类似效果。用户通过拖曳可以将页面上的元素放置到回收站中删除。效果如图 3-1 所示。

（a）页面初始加载效果　（b）拖动文件 2 的过程　（c）文件 2 被删除后的效果

图 3-1　仿回收站效果示意图

图 3-1 以删除文件 2 为例展示了对文件 2 拖动与删除的全过程。其他几个文件的操作效果与之完全相同，这里不再重复举例。

3.1.1　界面设计

本节主要介绍仿回收站效果的页面布局，主要包括文件展示区域和回收站区域两个部分。

1．使用 标签划分区域

可以使用 标签区分两个不同的区域：① 文件展示区；② 回收站区。

相关 HTML5 代码片段如下：

```
<body>
    <h3>HTML5拖放API之回收站效果</h3>
    <hr />
    <!--文件展示区域-->
    <div id="container"></div>
    <!--回收站区域-->
    <div id="recycle"></div>
</body>
```

分别为这两个区域的 标签定义 id 名称为 container 和 recycle，以便后续在 CSS 的 ID 选择器中使用。

此时还需要 CSS 文件辅助渲染样式，因此在本地 css 文件夹中创建 recycle.css 文件，并在<head>首尾标签中声明对 CSS 文件的引用。相关 HTML5 代码片段如下：

```
<head>
    <meta charset="utf-8">
    <title>HTML5拖放API之回收站效果</title>
    <link rel="stylesheet" href="css/recycle.css">
</head>
```

在 CSS 文件中为文件展示区域的 元素设置样式如下。

- 边框：1 像素宽的实线边框；
- 尺寸：宽 300 像素、高 250 像素；
- 浮动：向左浮动。

相关 CSS 代码片段如下：

```
/*设置用于放置文件夹的区域样式*/
div#container{
    border: 1px solid;
    width: 300px;
    height: 250px;
    float: left;
}
```

在 CSS 文件中为回收站区域的 元素设置样式如下。

- 尺寸：宽 200 像素、高 200 像素；
- 浮动：向左浮动；
- 文本：居中对齐；
- 背景：使用图片背景，素材来源于本地 image 目录下的 recycle.jpg 文件；
- 边距：各边的外边距为 30 像素。

相关 CSS 代码片段如下：

```
/*设置回收站样式*/
div#recycle {
    width: 200px;
    height: 200px;
    float: left;
    text-align: center;
    background: url(../image/recycle.jpg) no-repeat;
    margin: 30px;
}
```

目前尚未在文件展示区域内部添加示例文件，等待接下来补充。

2．使用 标签制作示例文件

为测试文件拖曳与回收的效果，在 id="container"的 元素内部继续使用 元素添加 4 个示例文件。相关 HTML5 代码片段修改后如下：

```
<!--文件展示区域-->
<div id="container">
    <div class="folder">文件1</div>
    <div class="folder">文件2</div>
    <div class="folder">文件3</div>
    <div class="folder">文件4</div>
</div>
```

为这 4 个 标签设置相同的 class 名称 folder，以便在 CSS 中使用类选择器统一设置样式效果。

在 CSS 文件中为 class="folder"的 标签设置统一样式如下。

- 文本：居中对齐；
- 浮动：向左浮动；
- 边距：各边的外边距为 20 像素；
- 背景：使用图片背景，素材来源于本地 image 目录下的 folder.png 文件；
- 尺寸：宽 100 像素、高 80 像素，行高 80 像素。

相关 CSS 代码片段如下：

```
/*设置文件夹样式*/
.folder{
    text-align: center;
    float: left;
    margin: 20px;
    background: url(../image/folder.png) no-repeat;
    width: 100px;
    height: 80px;
    line-height: 80px;
}
```

此时界面设计部分就全部完成了，运行后在浏览器中显示的效果如图 3-2 所示。

下面介绍如何使用 HTML5 拖曳 API 技术实现文件夹的拖曳和删除的效果。

图 3-2　仿回收站效果的界面设计完成图

3.1.2　文件拖曳与回收功能的实现

1．文件拖曳的实现

文件拖曳的实现比较简便，为 4 个用于显示文件夹图标的 元素添加 draggable 属性并将属性值设置为 true 即可。HTML5 相关代码片段修改后如下：

```
<!--文件展示区域-->
<div id="container">
    <div class="folder" draggable="true">文件1</div>
    <div class="folder" draggable="true">文件2</div>
    <div class="folder" draggable="true">文件3</div>
    <div class="folder" draggable="true">文件4</div>
</div>
```

此时文件拖曳已经可以实现了。以文件 2 为例，该文件被拖曳的运行效果如图 3-3 所示。

图 3-3　文件 2 被拖曳的效果图

由于目前尚未设置可放置区域，因此该文件只可以被拖曳，还无法放置到指定区域。接下来将介绍如何实现设置可放置区域。

2．将回收站区域设置为可放置区域

将元素设置为可放置区域需要添加 ondragover 事件。本例需要将回收站区域设置为可放置区域，因此在 id="recycle"的 元素中添加 ondragover 事件，并且设置自定义名称的回调函数 allowDrop(event)。

HTML5 相关代码片段修改后如下：

```
<!--回收站区域-->
<div id="recycle" ondragover="allowDrop(event)"></div>
```

在 JavaScript 中添加 allowDrop()函数，并且使用 preventDefault()方法解禁当前元素为可放置元素。相关 JavaScript 代码如下：

```
//ondragover事件回调函数
function allowDrop(ev) {
    //解禁当前元素为可放置被拖曳元素的区域
    ev.preventDefault();
}
```

此时文件拖曳到回收站区域上方时将被允许放置。以文件 2 为例，该文件被拖曳到回收站区域的运行效果如图 3-4 所示。

图 3-4　文件 2 在回收站区域可被放置的效果图

由图 3-4 可见，当文件 2 被拖曳到回收站区域上方时，原先显示的禁止符号消失。此时鼠标指针恢复正常指针样式，表示允许在当前区域放置被拖曳的元素。接下来将介绍如何实现文件拖曳到回收站区域放置可直接被删除的效果。

3．回收功能的实现

文件的删除需要依靠拖曳过程中数据的传递来实现。解决方案是在拖曳文件时传递当前元素的 id 名称，然后在回收站区域放置元素时根据被拖曳元素的 id 名称进行元素的删除。

首先为这 4 个用于展示文件的 元素分别设置不同的 id 名称以示区别。

相关 HTML5 代码片段修改后如下：

```
<!--文件展示区域-->
<div id="container">
    <div id="file1" class="folder" draggable="true">文件1</div>
    <div id="file2" class="folder" draggable="true">文件2</div>
    <div id="file3" class="folder" draggable="true">文件3</div>
    <div id="file4" class="folder" draggable="true">文件4</div>
</div>
```

当前这 4 个 元素的 id 名称分别设置为 file1～file4，开发者也可以根据实际情况修改成其他自定义 id 名称。

然后为这 4 个 元素添加 ondragstart 事件，并且设置自定义名称的回调函数 drag(event)用于传递被拖曳元素的 id 名称。相关 HTML5 代码片段修改后如下：

```
<!--文件展示区域-->
<div id="container">
    <div id="file1" class="folder" draggable="true" ondragstart="drag(event)">
    文件1</div>
    <div id="file2" class="folder" draggable="true" ondragstart="drag event)">
    文件2</div>
    <div id="file3" class="folder" draggable="true" ondragstart="drag(event)">
    文件3</div>
    <div id="file4" class="folder" draggable="true" ondragstart="drag(event)">
    文件4</div>
</div>
```

在 JavaScript 中添加 drag()函数，并且使用 DataTransfer 对象中的 setData()方法设置传递的数据为当前元素的 id 名称。相关 JavaScript 代码如下：

```
//ondragstart事件回调函数
function drag(ev) {
    //设置传递的内容为被拖曳元素的id名称，数据类型为纯文本类型
    ev.dataTransfer.setData("text/plain", ev.target.id);
}
```

为回收站区域的 元素添加 ondrop 事件，并且设置自定义名称的回调函数 drop(event)用于获取被放置元素的 id 名称。相关 HTML5 代码片段修改后如下：

```
<!--回收站区域-->
<div id="recycle" ondragover="allowDrop(event)" ondrop="drop(event)">
</div>
```

在 JavaScript 中添加 drop()函数，并且使用 DataTransfer 对象中的 getData()方法获取传递的数据，即当前被放置的元素 id 名称。然后根据 id 名称获取被放置的元素对象，并在文件展示区域使用 removeChild()方法删除该元素对象。相关 JavaScript 代码如下：

```
//ondrop事件回调函数
    function drop(ev) {
        //解禁当前元素为可放置被拖曳元素的区域
        ev.preventDefault();
        //获取当前被放置的元素id名称
        var id = ev.dataTransfer.getData("text");
        //根据id名称获取元素对象
```

```
        var folder = document.getElementById(id);
        //获取文件夹区域并删除该元素对象
        document.getElementById("container").removeChild(folder);
    }
```

以文件 2 为例，该文件被拖曳到回收站区域并删除的最终效果如图 3-5 所示。

图 3-5　文件 2 在回收站区域被删除的效果图

由图 3-5 可见，此时在文件展示区域只显示剩下的 3 个文件夹图标，放入回收站区域的元素已经彻底被删除。至此仿回收站效果的功能已经全部实现。

3.1.3　完整代码展示

HTML5 完整代码如下：

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <meta charset="utf-8">
5.          <title>HTML5拖放API之回收站效果</title>
6.          <link rel="stylesheet" href="css/recycle.css">
7.      </head>
8.      <body>
9.          <h3>HTML5拖放API之回收站效果</h3>
10.         <hr />
11.         <div id="container">
12.             <div id="file1" class="folder" draggable="true" ondragstart=
                "drag(event)">
13.                 文件1
14.             </div>
15.             <div id="file2" class="folder" draggable="true" ondragstart=
                "drag(event)">
16.                 文件2
17.             </div>
18.             <div id="file3" class="folder" draggable="true" ondragstart=
                "drag(event)">
19.                 文件3
```

```
            </div>
            <div id="file4" class="folder" draggable="true" ondragstart=
            "drag(event)">
                文件4
            </div>
        </div>
        <div id="recycle" ondragover="allowDrop(event)" ondrop="drop
        (event)"></div>
        <script>
            //ondragstart事件回调函数
            function drag(ev) {
                //设置传递的内容为被拖曳元素的id名称，数据类型为纯文本类型
                ev.dataTransfer.setData("text/plain", ev.target.id);
            }
            //ondragover事件回调函数
            function allowDrop(ev) {
                //解禁当前元素为可放置被拖曳元素的区域
                ev.preventDefault();
            }
            //ondrop事件回调函数
            function drop(ev) {
                //解禁当前元素为可放置被拖曳元素的区域
                ev.preventDefault();
                //获取当前被放置的元素id名称
                var id = ev.dataTransfer.getData("text");
                //根据id名称获取元素对象
                var folder = document.getElementById(id);
                //获取文件夹区域并删除该元素对象
                document.getElementById("container").removeChild(folder);
            }
        </script>
    </body>
</html>
```

CSS 完整代码如下：

```
/*设置用于放置文件夹的区域样式*/
div#container{
    border: 1px solid;
    width: 300px;
    height: 250px;
    float: left;
}
/*设置文件夹样式*/
.folder{
    text-align: center;
    float: left;
    margin: 20px;
    background: url(../image/folder.png) no-repeat;
    width: 100px;
    height: 80px;
    line-height: 80px;
}
/*设置回收站样式*/
div#recycle {
```

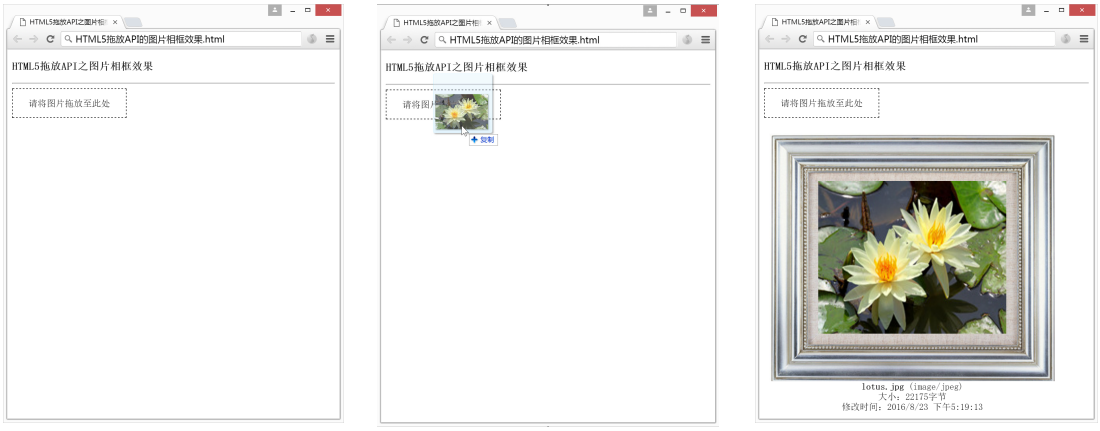
```
    width: 200px;
    height: 200px;
    float: left;
    text-align: center;
    background: url(../image/recycle.jpg) no-repeat;
    margin: 30px;
}
```

3.2 图片相框展示的设计与实现

【例 3-2】 基于 HTML5 拖曳 API 的图片相框展示的设计与实现

背景介绍：目前市面上的一些修图工具软件带有自动为图片添加不同款式相框的功能，用户可以选择本地图片文件然后为其添加相框效果。

功能要求：使用 HTML5 拖放 API 相关技术在网页上实现为指定图片自动生成图片相框的效果。用户通过拖曳将本地的图片文件放置到页面上的指定区域即可在页面上自动生成带有相框效果的图片展示。效果如图 3-6 所示。



（a）页面初始加载效果　　（b）本地图片文件的拖曳　　（c）自动生成相框效果

图 3-6　图片相框展示的效果示意图

将本地的图片文件拖曳到页面上的指定区域进行放置，即可生成带有相框样式的图片展示效果。由图 3-6 可见，生成的最终效果还包括原始图片的名称、类型、大小和修改时间等相关信息。

3.2.1 界面设计

本节主要介绍示例项目的页面布局设计，主要包括本地文件放置区域和带有相框图片的展示区域两个部分。

可以使用 标签区分这两个区域，相关 HTML5 代码片段如下：

```
<body>
    <h3>HTML5拖放API之图片相框效果</h3>
    <hr />
    <!--可放置文件区-->
```

```
    <div id="recycle">请将图片拖放至此处</div>
    <br />
    <!--带有相框的图片展示区-->
    <div id="output"></div>
</body>
```

分别为这两个区域的 标签定义 id 名称为 recycle 和 output，以便后续在 CSS 的 ID 选择器中使用。

此时还需要 CSS 文件辅助渲染样式，因此在本地 css 文件夹中创建 photoframe.css 文件，并在<head>首尾标签中声明对 CSS 文件的引用。相关 HTML5 代码片段如下：

```
<head>
    <meta charset="utf-8">
    <title>HTML5拖放API之图片相框效果</title>
    <link rel="stylesheet" href="css/photoframe.css">
</head>
```

在 CSS 文件中为可放置区域的 元素设置样式如下。

- 尺寸：宽 200 像素、高 50 像素；
- 边框：1 像素宽的虚线边框；
- 文本：居中对齐；
- 行高：单行高度为 50 像素。

相关 CSS 代码片段如下：

```
/*设置可放置区域样式*/
#recycle {
    width: 200px;
    height: 50px;
    border: 1px dashed;
    text-align: center;
    line-height: 50px;
}
```

在 CSS 文件中为图片展示区域的 元素设置样式如下。

- 浮动：向左浮动。
- 边距：各边的外边距为 10 像素。
- 文本：居中对齐。
- 尺寸：宽度为 500 像素。

相关 CSS 代码片段如下：

```
/*设置带有相框的图片展示区域样式*/
#output {
    float: left;
    margin: 10px;
    text-align: center;
    width: 500px;
}
```

此时界面设计部分全部完成，运行后在浏览器中显示的效果如图 3-7 所示。

下面介绍如何使用 HTML5 拖曳 API 技术实现图片文件的拖曳与相框自动生成效果。

图 3-7　图片相框展示的界面设计效果图

3.2.2　相框自动生成功能的实现

1．可放置区域的实现

将元素设置为可放置区域需要添加 ondragover 事件。本示例需要在 id="recycle"的 元素中添加 ondragover 事件，并且设置自定义名称的回调函数 allowDrop(event)。

HTML5 相关代码片段修改后如下：

```
<!--可放置文件区-->
<div id="recycle" ondragover="allowDrop(event)">请将图片拖放至此处</div>
```

在 JavaScript 中添加 allowDrop()函数，并且使用 preventDefault()方法解禁当前元素为可放置元素。相关 JavaScript 代码如下：

```
//ondragover事件回调函数
function allowDrop(ev) {
    //解禁当前元素为可放置被拖曳元素的区域
    ev.preventDefault();
}
```

此时文件拖曳到 id="recycle"的 元素上方时将被允许放置。本地图片文件被拖曳到可放置区域的运行效果如图 3-8 所示。

图 3-8　本地图片文件可被放置的效果图

接下来将介绍如何在页面上生成带有相框的图片效果。

2．生成带有相框的图片效果

为 id="recycle"的 元素添加 ondrop 事件，并且设置自定义名称的回调函数 fileDrop(event)。当用户在指定区域放置图片文件时会触发 fileDrop()函数。

HTML5 相关代码片段修改后如下：

```
<!--可放置文件区-->
<div id="recycle" ondragover="allowDrop(event)" ondrop="fileDrop(event)">
    请将图片拖放至此处
</div>
```

在 JavaScript 中添加 fileDrop()函数，并且使用 preventDefault()方法解禁当前元素为可放置元素。然后获取 id="output"的 元素对象，并使用其 innerHTML 属性清空内容。相关 JavaScript 代码如下：

```
//ondrop事件回调函数
function fileDrop(e) {
    //解禁当前元素为可放置被拖曳元素的区域
    e.preventDefault();

    //获取图片展示区域对象output
    var output = document.getElementById("output");
    //将图片展示区域的内容清空
    output.innerHTML = "";
}
```

用该方法清空 output 对象的内容主要是为了在重复放入图片文件时可以清除上一次的图片内容。如果本次为首次放置文件，则 output 对象的内容原本就为空。

在 JavaScript 的 fileDrop()方法中添加代码。由于被拖放的文件允许是一个或者多个，因此使用 Datatransfer 对象中的 files 属性获取同时被拖放的文件信息。然后使用 for 循环语句遍历每一个被放置的本地文件。新增内容的 fileDrop()方法的代码片段如下：

```
//ondrop事件回调函数
function fileDrop(e) {
    …

    //获取从本地拖曳放置的文件对象数组files
    var files = e.dataTransfer.files;

    //使用for循环遍历同时被拖曳并放置到指定区域的所有文件
    for (var i = 0,f; f = files[i]; i++) {
        //待补充代码
    }
}
```

在 JavaScript 中使用 document.createElement()方法动态创建图片元素，将其 src 属性设置为被拖放的本地图片文件地址，并分别设置其宽 330 像素、高 270 像素。新增内容的 fileDrop()方法的代码片段如下：

```
//ondrop事件回调函数
```

```
function fileDrop(e) {
    ⋮
    //使用for循环遍历同时被拖曳并放置到指定区域的所有文件
    for (var i = 0,f; f = files[i]; i++) {
        //创建带有相框的图片
        //获取当前图片文件的URL来源
        var imgURL = window.webkitURL.createObjectURL(f);
        //创建图片对象img
        var img = document.createElement("img");
        //设置图片对象img的src属性为当前图片文件的URL地址
        img.setAttribute("src", imgURL);
        //设置图片对象img的宽度为330像素
        img.setAttribute("width", "330");
        //设置图片对象img的高度为270像素
        img.setAttribute("height", "270");

        //待补充代码
    }
}
```

此时用户拖曳并放置的每一张图片都将产生一个宽 330 像素、高 270 像素的<img>元素。

接下来在 for 循环中使用 document.createElement()方法动态创建 元素，将其背景图片设置为相框样式，并将图片元素添加到相框元素中。

新增内容的 fileDrop()方法的代码片段如下：

```
//ondrop事件回调函数
function fileDrop(e) {
    ⋮
    //使用for循环遍历同时被拖曳并放置到指定区域的所有文件
    for (var i = 0,f; f = files[i]; i++) {
        //创建带有相框的图片
        ⋮
        //设置相框对象photo
        var photo = document.createElement("div");
        //为相框对象添加class="photoframe"，以加载相框背景图片
        photo.setAttribute("class","photoframe");
        //将图片添加到相框对象中
        photo.appendChild(img);

         //待补充代码
    }
}
```

其中 setAttribute("class","photoframe")语句表示为用于展示相框的 元素添加 class="photoframe"属性。该 class 名称可自定义，并且需要在 CSS 文件中设置相关样式内容。

在 CSS 文件中使用类选择器为用于展示相框的 元素设置样式如下。

- 背景：使用图片背景，素材来源于本地 image 目录下的 photoframe.jpg 文件。
- 尺寸：宽 500 像素，高 438 像素。
- 文本：居中对齐。

- 浮动：向左浮动。

相关 CSS 代码片段如下：

```
/*设置图片相框效果样式*/
.photoframe {
    background: url(../image/photoframe.jpg) no-repeat;
    width: 500px;
    height: 438px;
    text-align: center;
    float: left;
}
```

此时图片尚不能正常居中显示。为了快速实现图片在垂直方向上居中对齐的效果，需要在其后面添加一个宽度为 0、高度为 100%的图片 2。图片 2 不占空间位置，仅用于辅助图片居中效果。

新增内容的 fileDrop()方法的代码片段如下：

```
//ondrop事件回调函数
function fileDrop(e) {
    ⋮
    //使用for循环遍历同时被拖曳并放置到指定区域的所有文件
    for (var i = 0,f; f = files[i]; i++) {
        //创建带有相框的图片
        ⋮
        //创建图片对象img2
        var img2 = document.createElement('img');
        //设置图片对象img2的class="block"
        img2.setAttribute("class", "block");
        //将图片2也添加到相框元素中
        photo.appendChild(img2);

        //添加相框和图片效果
        output.appendChild(photo);

        //待补充代码
    }
}
```

其中，setAttribute("class","block")语句表示为图片 2 添加 class= "block"属性。该 class 名称可自定义，并且需要在 CSS 文件中设置相关样式内容。

在 CSS 文件中使用类选择器为图片 2 设置样式如下：

```
/*设置图片2的样式*/
.block {
    width: 0px;
    height: 100%;
}
```

此时图片 2 可以帮助需要显示的图片实现垂直居中效果。运行效果如图 3-9 所示。

接下来将介绍如何在页面上显示图片文件的相关信息，包括原始图片的名称、类型、

大小和修改时间等信息。

3．显示图片文件信息

修改 JavaScript 中的 fileDrop()方法，在其中的 for 循环语句内部继续添加用于显示图片信息的相关代码。文件的相关信息均来源于 files 数组中的每一个文件对象。

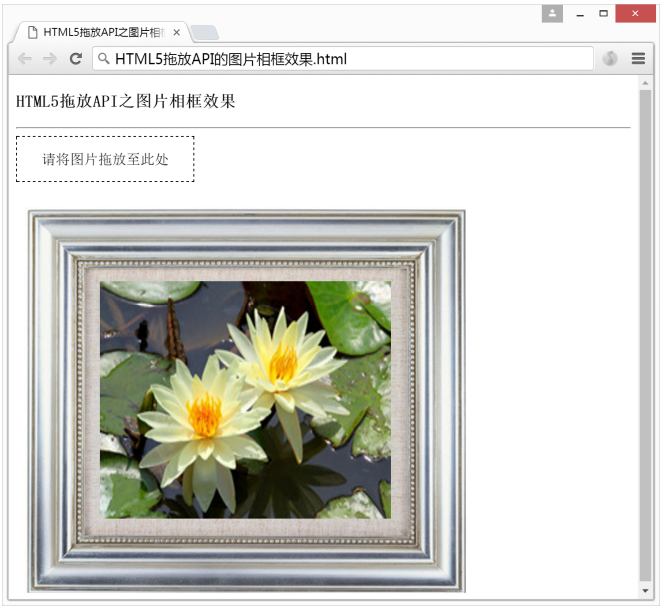


图 3-9　相框自动生成的效果图

新增内容的 fileDrop()方法的代码片段如下：

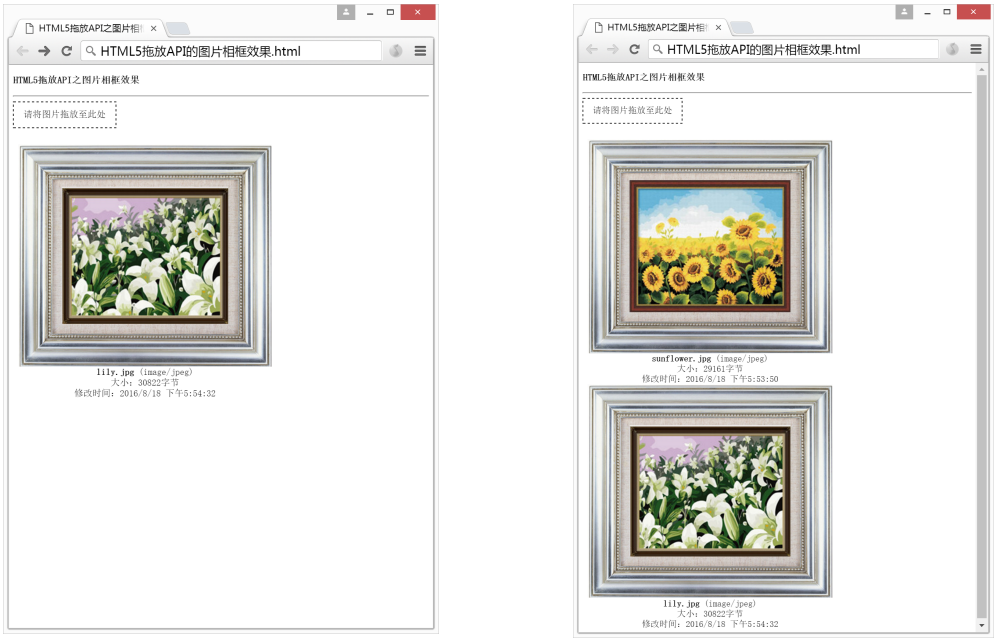
```
//ondrop事件回调函数
function fileDrop(e) {
    ⋮
    //使用for循环遍历同时被拖曳并放置到指定区域的所有文件
    for (var i = 0,f; f = files[i]; i++) {
        //创建带有相框的图片
        ⋮

        //创建图片下方的状态信息栏
        //使用div元素创建状态信息栏status
        var status = document.createElement('div');
        //获取当前文件的最新修改日期
        var lastModified = f.lastModifiedDate;
        //修改当前文件的最新修改日期的描述格式
        var lastModifiedStr = lastModified ? lastModified.toLocaleDateString()
        + ' ' + lastModified.toLocaleTimeString() : 'n/a';
        //设置图片下方状态信息栏的描述内容
        status.innerHTML = '<strong>' + f.name + '</strong> (' + (f.type ||
        'n/a') + ')<br>大小：' + f.size + '字节<br>修改时间：' + lastModifiedStr;

        //添加文件描述
        output.appendChild(status);
```

```
        }
    }
```

分别尝试拖放一个和多个本地图片文件，其效果如图 3-10 所示。



（a）单个图片文件的显示效果　　（b）多个图片文件的显示效果

图 3-10　为单个或多个图片生成相框的效果图

由图 3-10 可见，本例项目也支持本地图片文件的批量拖放，并能够分别为每一张图片生成相框效果。至此图片相框展示的功能已经全部实现。

3.2.3　完整代码展示

HTML5 完整代码如下：

```
1.   <!DOCTYPE html>
2.   <html>
3.       <head>
4.           <meta charset="utf-8">
5.           <title>HTML5拖放API之图片相框效果</title>
6.           <link rel="stylesheet" href="css/photoframe.css">
7.       </head>
8.       <body>
9.           <h3>HTML5拖放API之图片相框效果</h3>
10.          <hr />
11.          <!--可放置文件区-->
12.          <div id="recycle" ondragover="allowDrop(event)" ondrop="fileDrop
             (event)">
13.              请将图片拖放至此处
14.          </div>
15.          <br />
16.          <!--带有相框的图片展示区-->
```

```
17.         <div id="output"></div>
18.         <script>
19.             //ondragover事件回调函数
20.             function allowDrop(ev) {
21.                 //解禁当前元素为可放置被拖曳元素的区域
22.                 ev.preventDefault();
23.             }
24.             //ondrop事件回调函数
25.             function fileDrop(e) {
26.                 //解禁当前元素为可放置被拖曳元素的区域
27.                 e.preventDefault();
28.
29.                 //获取图片展示区域对象output
30.                 var output = document.getElementById("output");
31.                 //将图片展示区域的内容清空
32.                 output.innerHTML = "";
33.
34.                 //获取从本地拖曳放置的文件对象数组files
35.                 var files = e.dataTransfer.files;
36.
37.                 //使用for循环遍历同时被拖曳并放置到指定区域的所有文件
38.                 for (var i = 0,f; f = files[i]; i++) {
39.                     //创建带有相框的图片
40.                     //获取当前图片文件的URL来源
41.                     var imgURL = window.webkitURL.createObjectURL(f);
42.                     //创建图片对象img
43.                     var img = document.createElement("img");
44.                     //设置图片对象img的src属性为当前图片文件的URL地址
45.                     img.setAttribute("src", imgURL);
46.                     //设置图片对象img的宽度为330像素
47.                     img.setAttribute("width", "330");
48.                     //设置图片对象img的高度为270像素
49.                     img.setAttribute("height", "270");
50.
51.                     //设置相框对象photo
52.                     var photo = document.createElement("div");
53.                     //为相框对象添加class="photoframe"，以加载相框背景图片
54.                     photo.setAttribute("class", "photoframe");
55.                     //将图片添加到相框对象中
56.                     photo.appendChild(img);
57.
58.                     //创建图片对象img2
59.                     var img2 = document.createElement("img");
60.                     //设置图片对象img2的class="block"
61.                     img2.setAttribute("class", "block");
62.                     //将图片2也添加到相框元素中
63.                     photo.appendChild(img2);
64.
65.                     //添加相框和图片效果
66.                     output.appendChild(photo);
67.
68.                     //创建图片下方的状态信息栏
69.                     //使用div元素创建状态信息栏status
70.                     var status = document.createElement("div");
71.                     //获取当前文件的最新修改日期
72.                     var lastModified = f.lastModifiedDate;
```

```
                    //修改当前文件的最新修改日期的描述格式
                    var lastModifiedStr = lastModified ? lastModified.
                    toLocaleDateString() + ' ' + lastModified.toLocale
                    TimeString() : 'n/a';
                    //设置图片下方状态信息栏的描述内容
                    status.innerHTML = '<strong>' + f.name + '</strong>
                    (' + (f.type || 'n/a') + ')<br>大小：' + f.size +
                     '字节<br>修改时间：' + lastModifiedStr;

                    //添加文件描述
                    output.appendChild(status);
                }
            }
        </script>
    </body>
</html>
```

完整的 CSS 文件代码如下：

```
/*设置可放置区域样式*/
#recycle {
    width: 200px;
    height: 50px;
    border: 1px dashed;
    text-align: center;
    line-height: 50px;
}
/*设置图片相框效果样式*/
.photoframe {
    background: url(../image/photoframe.jpg) no-repeat;
    width: 500px;
    height: 438px;
    text-align: center;
    float: left;
}
/*设置图片在垂直方向上居中显示*/
img {
    vertical-align: middle;
}
/*设置图片2的样式*/
.block {
    width: 0px;
    height: 100%;
}
/*设置带有相框的图片展示区域样式*/
#output {
    float: left;
    margin: 10px;
    text-align: center;
    width: 500px;
}
```